Learning Components for A Question-Answering System

Dan Roth
danr@cs.uiuc.edu
Gio Kao Kao, Xin Li, Ramya Nagarajan,
Vasin Punyakanok, Nick Rizzolo, Wen-tau Yih
Department of Computer Science

Cecilia Ovesdotter Alm, Liam Gerard Moran
Department of Linguistics

University of Illinois at Urbana-Champaign

Abstract

We describe a machine learning approach to the development of several key components in a question answering system and the way they were used in the UIUC QA system.

A unified learning approach is used to develop a part-of-speech tagger, a shallow parser, a named entity recognizer and a module for identifying a question's target. These components are used in analyzing questions, as well as in the analysis of selected passages that may contain the sought after answer.

The performance of the learned modules seems to be very high, (e.g., mid 90% for identifying noun phrases in sentences), though evaluating those on a large number of passages proved to be time consuming. Other components of the system, a passage retrieval module and an answer selection module, were put together in an ad-hoc fashion and significantly affected the overall performance. We ran the system only over about 60% of questions, answering a third of them correctly.

1 Introduction

The QA system described in this paper is developed as a platform for studying and experimenting with a unified method for learning, knowledge representation and inference, required to perform knowledge intensive natural language based inferences.

Our working assumption is that a robust and accurate question answering system will depend on a large number of predictors. These will be used at many levels of the process and will support a variety of functions, from knowledge acquisition to decision making and integration of information sources. Along with these, there needs to be a knowledge representation support that allows, for example, to keep track of predictions as input to higher level predictors and maintain a coherent representation of a question or a story; and, there needs to be an ability to use the outcomes of lower level predictions to make inferences that use of several of these predictors along with some constraints, e.g., those that are implied by the questions.

The system developed here makes some preliminary steps in these directions by putting forward a suggestion for a few of the learning components and placing them, for evaluation purposes, within a question answering system. We describe several key components in a question answering system, developed within a unified learning approach that makes use of a relational representation language. Some of the components presented also incorporate domain and task specific constraints as part of a general scheme for inference with outcomes of classifiers that, we believe, could have a more general use.

The learning components used here include a POS tagger, a shallow parser and a named entity recognition module (which is essentially the same module, only trained differently) and a module for identifying a question's target. These components were used in analyzing questions, as well as in the analysis of selected passages that may contain the sought after answer.

This project started as a summer project in early June, 2001. The reliance on learning methods allowed us to put together a system for this task in about six weeks. Needless to say, there are several important components that are still missing. The main part missing from our current approach, due to lack of time, is a learned module for selecting an appropriate answer given a candidate passage and the constraints identified in the question analysis. During the work on this task we realized that, given the vast amount of text available, there is almost always a "simply structured" correct answer among the large number of correct answers that exist in a corpus. This makes the task very different from the *story comprehension* task [5] because simple heuristics that rely on the existence of a simply structured answer can already give reasonable results; due to lack of time, we resorted to these in the current system. A second significant component that is missing is an information retrieval module, which was not in the focus of our study. We used the documents retrieved by TREC and a simple-minded approach to focus on candidate passages within those documents.

This report describes the main learning components (Sec. 2), how these are used in our system (Sec. 3), and some preliminary evaluation of the learning components and the system (sec. 4). We conclude with some questions that pertain to our approach and comments on future plans.

2 Learning Components for a QA system

A robust and accurate question answering system depends on a large number of classifiers that will be used at different levels of the process and will support a variety of functions, from knowledge acquisition to decision making and the integration of information sources to yield robust decisions.

In this project, we used a unified methodology to develop and study a few learning components that we believe are necessary. Each of our learning modules consists of a stage of generating expressive relational features (that could rely on previously learned predictors or knowledge acquired otherwise), learning using a network of linear classifiers over these, and an inference process that makes use of the outcome of the classifiers to make decisions that relate some domain and task specific constraints.

This report describes four learning components that are used in the current QA system. All components make use of the same learning approach and tools. Although several of the components are built on the results of previously learned predictors, learning is always done one stage at a time.

Learning is done using the SNoW learning architecture. SNoW [1, 11] is a multi-class classifier that is specifically tailored for learning in domains in which the potential number of information sources (features) taking part in decisions is very large, of which NLP is a principal example. It works by learning a sparse network of linear functions over a pre-defined or incrementally learned feature space. Typically, SNoW is used as a classifier, and predicts using a winner-take-all mechanism over the activation value of the target classes. However, in addition to the prediction, it provides a reliable confidence level in the prediction, which enables its use in an inference algorithm that combines predictors to produce a coherent inference. (See the descriptions of the components for pointers to the details on how SNoW is being used in each case.)

The input to SNoW is provided by a feature extraction stage. In this stage we use a knowledge representation language [2, 12] to generate expressive relational features over a collection of predicates. These predicates can be computed directly from the input data (e.g., words in a given sentence), previously learned predicates (e.g., pos tags) or predicates that are acquired in other ways (e.g., a semantic class of a word). Predicates of all these types are used in the learning components described below. A programmer needs only to define a small number of feature *types* (RGFs, in [2, 12]) believed to be necessary for representing the classifier, and many features of this type will be generated in a data driven manner, as input sentences are observed. In all problems discussed below, the number of potential features generated will be very large, but the learning approach is capable of learning in the presence of a large number of potential features.

Several of the components described below already include ways of interaction among classifiers. These include the sequential model [3], used for the pos tagger, as well as the question classifier and the CSCL shallow parser [10], used for shallow parsing and name entity recognition.

2.1 Part-Of-Speech tagger

The POS tagger used here is the one developed in [3]. This is a SNoW based pos tagger that makes use of a sequential model of classification to restrict the number of competing classes (pos tags) while maintaining, with high probability, the presence of the true outcome in the candidate set. The same method is used to give pos tags to both known and unknown words. Overall, as shown in [3], it achieves state-of-the-art results on this task and is significantly more efficient than other part-of-speech taggers. The tagger and a demo of it are available at `http://L2R.cs.uiuc.edu/~cogcomp`

2.2 Sentence Analysis

Our sentence analysis makes use of an *inference with classifiers* paradigm as one general method for identification of phrases in sentences. The same method, a SNoW-based CSCL parser [10, 9], is used by both the shallow parser and the name entity analyzer.

In CSCL (constraint satisfaction with classifiers), SNoW is used to learn several different word level classifiers – each detects the beginning or end of a phrase of some type (noun phrase, verb phrase, a location phrase, etc.). These classifiers are learned as a function of the words and pos tags in the context of the target word. The outcomes of these classifiers are then combined to a sentence level decision in a way that satisfies some constraints – *non-overlapping* constraints in this case – using an efficient constraint satisfaction mechanism that makes use of the confidence in the classifier's outcomes. This method can be used to identify the phrases of one type (as in [10, 9]) or of several different types at the same time [8]. We use two instantiations of this method below, with different training data.

2.2.1 Shallow parsing

Shallow parsing, also known as text chunking, is the task of identifying phrases, possibly of several types, in natural language sentences. It is simpler, conceptually and computationally, than full parsing, but still provides fundamental sentence structure information such as noun phrases and verb phrases. An additional advantage from a learning perspective is that limited training data can still be used to induce a shallow parser for the type of information available.

The shallow parser used here is based on [10]. As mentioned above, for each type of phrase, two learned classifiers are used, one learns to identify the beginning of the phrase, and the other its end. The final prediction is made using a constraint satisfaction based inference, which takes into account constraints such as "Phrases do not overlap".

In Question Analysis, we use this module to identify three types of phrases: noun-phrases, verb-phrases and prepositional-phrases. The definitions of these phrases follow those in the text chunking shared task in CoNLL-2000 [7]. When analyzing retrieved passages, in order to save processing time, only noun-phrases and verb-phrases are identified. Below are some examples to the type of processing provided by this module.

Question: *What was the name of the first Russian astronaut to do a space walk?*

> [NP What] [VP was] [NP the name] [PP of] [NP the first Russian astronaut]
> [VP to do] [NP a spacewalk]

Sentence: *The broad-shouldered but paunchy Leonov, who in 1965 became the first man to walk in space, signed autographs.*

> [NP The broad-shouldered but paunchy Leonov] , [NP who] in [NP 1965] [VP became]
> [NP the first man] [VP to walk] in [NP space] , [VP signed] [NP autographs] .

The shallow parser and a demo of it are available at `http://L2R.cs.uiuc.edu/~cogcomp.html`

2.2.2 Recognizing named entity phrases

The named entity recognizer annotates various types of named entities. Unlike other common named entity recognition systems which only annotate proper nouns, dates, time, and other numerical values, our named entity recognizer attempts to further extend the scope of annotated categories and the definition of a "named entity".

In addition to annotating typical categories such as person, organization, location, time, money, date, and percentage, we add several more categories that are typically not proper nouns. Some of the additional categories are title, profession, event, holiday & festival, animal, plant, sport, medical, unit, etc. These extra categories provide more information than a typical entity recognition system and can be viewed more as a step toward semantic categorization. When relevant, we further sub-divide categories into more detailed subclasses (e.g. Location-City, Location-Country).

To achieve this named entity recognition task, we have combined machine learning techniques with a manually based knowledge acquisition process that we used to acquire categorized lists, as well as some specific rules that are used to integrate these. Our plan was to use the categorized list as additional annotation for training but, in the current system, we had enough data to train only of a few of the large categories.

For three major categories, those of person, location, and organization, the SNoW based CSCL approach [10] described above. Phrases of these types were annotated using the categorized lists we generated, and CSCL was used to learn a phrase recognizer for these types of phrases. In evaluation, the list-based process first annotates the data as suggested named entities, and then the classifier uses this suggestion, along with the context in the sentence, to provide a more accurate annotation of the data. Other, smaller categories, are tagged using the lists and some rules that incorporate special keywords and stop lists. By processing the data through several large categorized lists, we are able to annotate the remaining categories. A few of the categories are exhausted by a combination of lists and rules.

It is important to mention that, although we find CSCL a promising approach to this problem, there are several interesting problems that we still need to address here. The most important are the training data problem and that of defining categories (and perhaps hierarchies of) in a satisfactory manner.

2.3 Question Classification

Inspired by many questions answering systems in previous TREC Q/A tracks [4, 6, 13], we believe identifying the target of a question is an important step in an attempt to answer it correctly. When a system is aware of being asked a who-question, it can focus on names or titles as potential answers. However, our working assumption is that it is better to classify questions into finer categories and not rely solely on the head of the question (e.g. what, who, or when). This will enhanced the performance of the system, by allowing it to look for more specific and accurate potential answers. it may also allow for the development of different strategies for answer selection that depend on the fine classification of the question class.

We developed a learning approach to this problem that utilizes the sequential model idea [3] in learning a hierarchy of question classes. The question classes were organized into a two-layered hierarchy, that also allows us to tradeoff the accuracy of the classification with the concreteness of question classes. In particular, classes in the first layer are easier to predict, whereas classes in the second layer provide a more concrete specification of the target answers. We defined *eight* top classes and about *fifty* final classes in this hierarchy. The top classes we use are *Abbreviation*, *Abstract Entity*, *Concrete Entity*, *Description*, *Human*, *Location*, *Number*, *Other Entity*. The final classes include *color*, *language*, *animal*, *sport*, *definition*, *reason*, *city*, *country*, *age*, *date*, *speed*, and so on.

The question classifier is trained using Sequential Model and the SNoW learning architecture. The training set includes about $6,000$ questions, consisting of TREC-8 and TREC-9 questions, and other questions that we generated - all manually annotated. Features for the question classifier were generated using the FEX feature extractor [2] and used predicates that include information in the sentence (words and sentence length), previously learned predicates (pos tags, shallow parsing, named entity) and some semantic categorization information acquired using WordNet.

3 System Description

As shown in Fig.1, our QA system consists of three main modules, supported by the learning components described in last section. Using all the learning components, Question Analyzer extracts semantic and syntactic information from a question and stores it in question analysis records. Passage Retriever uses this information to extract relevant passages from the corresponding documents that are retrieved by the search engine. Given the question analysis record and relevant passages, Answer Selector analyzes the documents with the help of POS tagger, NE recognizer, shallow parser, and then finds answers.

3.1 Question Analysis

The goal of Question Analyzer is to transform questions into new representations, which provide further information to the other modules. Tasks that Question Analyzer performs include:

- Question Classification: deciding the types of potential answers by classifying question types. (supported by Question Classifier)

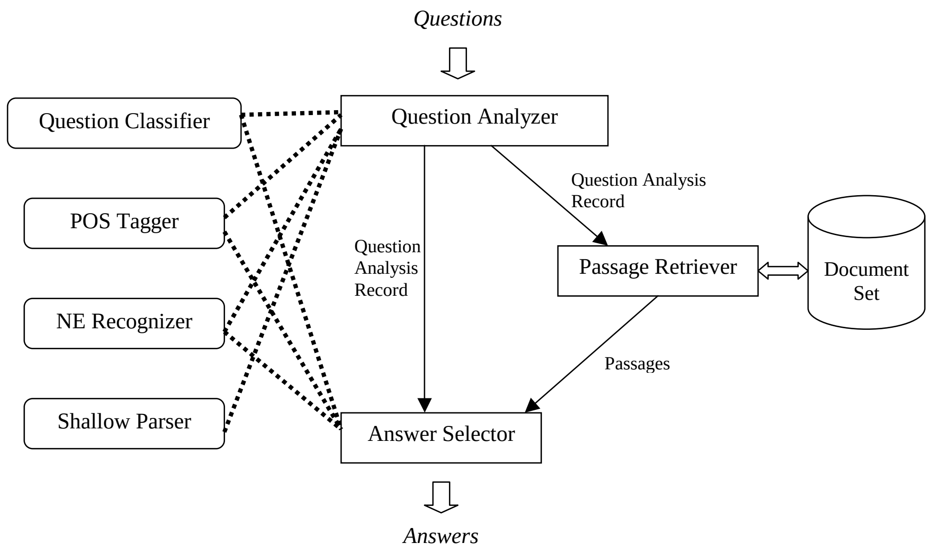

Figure 1: Top level system architecture

- Key Phrase Retrieval: finding keywords in the question.
- Semantic Representation: representing the semantic information from questions. Different types of questions may have different representations.

We describe the last two tasks as follows.

3.1.1 Key Phrase Retrieval

Key phrases are the query terms used in searching relevant documents and passages. In our system, there are seven types of key phrases extracted, including *Quotation*, *Named Entity*, *Noun*, *Verb*, *Extreme Adjective*, *Capitalized Word*, and *Unknown Word*.

Key phrases are extracted sequentially according to the same order. For instance, if a word in the question has been identified as part of a quotation, then it won't be taken into account as other key phrase types.

3.1.2 Semantic Representation

The purpose of acquiring some form of representation for a question is to locate and verify candidate answers. A search engine is used to obtain candidate documents containing search terms. However, the constraints of the semantic relation between those terms haven't been expressed in this process. Thus, we need a semantic representation to represent the real 'meaning' of a question so that we can locate the exact answer in a document and verify its correctness by comparing the representations of the question and an answer candidate.

Syntactic analysis is the underlying step for acquiring semantic representation. We apply POS tagging and shallow parsing to questions. Syntactic analysis can be somewhat easy and can achieve a high accuracy. The Name Entity recognizer is then used to get semantic tags for words and phrases in the question sentence.

It's not clear so far which form of representation is appropriate for this task and easy to acquire at the same time. But it's somewhat clear that we don't need very complicated logic representation, since the major application of it is matching rather than reasoning. As a beginning, we predefined a simple template to represent the knowledge in questions and use simple heuristics to get information defined in the template from questions and answers. Although it is relatively easy to convert a question to a simple format of semantic representation, it seems not enough.

Some fields in the template are: *Answer Entity Type*, *Action*, *Action Subject*, *Direct Object*, *Indirect Object*, *Target Description*, *Target Modifier*, *Action Modifier*, *Location*, *Time*, *Extreme Case*, and *Unit*.

3.2 Passage Retriever

To reduce the search scope of Answer Selector, Passage Retriever picks relevant passages from the 1000 documents that TREC provides for each question. Based on the key phrases extracted by Question Analyzer, Passage Retriever first filters out some documents, then retrieves relevant passages from the preserved documents. Key phrases are not directly used as query terms. Instead, they are expanded by selectively adding synonyms or related words according to the information in WordNet.

Criteria for relevant documents and relevant passages are described as follows.

Document Retrieval A document that is classified as relevant must have all the key phrases or their expansions in it.

Passage Extraction A passage is defined as a short paragraph which contains no more than five consecutive sentences. If all the key phrases or their expansions can be found in these five sentences, then the passage is considered as relevant and returned by Passage Retriever. However, if a key phrase is a full name of a person, then the last name or the first name are both treated as valid synonyms. This is because a document could mention a person's full name in the beginning, and then only use his first name or last name later.

3.3 Answer Selector

Given question analysis records, Answer Selector finds answers from extracted passages. It consists of the following three steps. Sentences in these passages are first analyzed syntactically and semantically by POS tagger, shallow parser, and name entity recognizer. Candidate answers are then located in these processed passages. Finally, each candidate answer is evaluated and ranked. The top five answers are extended or shrunk to satisfy the 50-byte length constraint and returned as final answers.

Although we believe that the robust inference procedure based on learning is the right way for choosing and verifying answers, the current version of our QA system uses only heuristic-based, ad-hoc procedure instead because of limited development time.

Decisions on locating candidate answers strongly rely on the results of the question classifier, name entity recognizer and shallow parser. For example, if a question asks for a person, a location, or some number, then only the phrases belonging to these name entity types will be treated as candidates. For other question classes, a noun-phrase or even a whole sentence will be picked as potential answers.

To rank all the candidate answers, we evaluate the confidence scores based on some heuristic rules. These rules generally test how closely candidate answers match the question in terms of keywords and phrases in different semantic fields. For instance, a candidate answer will get higher confidence if many of the nearby phrases contain or overlap *Target Modifier*, *Extreme Case*, or other semantic fields identified from the question.

4 Evaluation

4.1 Evaluation of Learning Components

We first present results on the performance of the learning components as stand alone modules.

4.1.1 POS

The test corpus for our POS tagger is taken from the Penn Treebank WSJ and Brown corpora. It consists of 280,000 words, of which 5,412 are unknown words (words that do not appear in the training corpus). For the known words, the accuracy of our POS tagger is 96.86%, which is slightly better than Brill's POS tagger (96.49%), but the speed is 3000 times faster. For the unknown words, we still have reasonably high accuracy (73.0%). More details of the evaluation can be found in [3].

4.1.2 Shallow Parsing

The evaluation of our shallow parser consists of two parts. The first is to compare it with other shallow parsers, and the second is to compare it with a full sentence parser.

To compare our shallow parser with others, we chose the data used in the chunking competition in CoNLL-2000 [7]. In this competition, a full parse tree is represented in a flat form. The goal in this case is to accurately predict a collection of 11 different types of phrases. The chunk types are based on the syntactic category part of the bracket label in the Treebank. Using exactly the same training and testing data, our shallow parser ranks among the top ones.

Additionally, we also demonstrate that by focusing only on the most significant syntactic information, shallow parsing is not only much faster, but can also achieve more accurate results than the full parser. We design several experiments and compare our shallow parser to Michael Collins' full parser, which is one of the most accurate full parsers. For tasks of phrase identification on both WSJ data and Switchboard data, our parser outperforms Collins' full parser in every experiment. The overall experimental results in terms of F_β value are shown in Table 1. For more details, please refer to [8].

Table 1: **Precision & Recall for phrase identification (chunking)** for the full and the shallow parser on the WSJ data. Results are shown for an average of 10 types of phrases as well as for two of the most common phrases, NP and VP.

	Full Parser			Shallow Parser		
	P	R	$F_{\beta=1}$	P	R	$F_{\beta=1}$
Avrg	91.71	92.21	91.96	93.85	95.45	94.64
NP	93.10	92.05	92.57	93.83	95.92	94.87
VP	86.00	90.42	88.15	95.50	95.05	95.28

4.1.3 Named Entity

Although our named entity recognizer is designed to tag many detailed sub-classes, to make a fair comparison, we test it on the benchmark dataset from MUC-7, which contains only three tags: *Person*, *Location*, and *Organization*. 2000 sentences are used as training data and 430 sentences are used as testing data. The recall-precision results are shown in Tab. 2.

Table 2: Precision & Recall for named entity recognition

	Recall	Precision	$F_{\beta=1}$
Overall	75.97	92.64	83.48
Person	68.50	93.98	79.24
Location	85.75	91.49	88.53
Organization	70.22	93.12	80.06

4.1.4 Question Classifier

We manually label all the 500 questions in TREC-10, and use them as testing data for our question classifier. The classifier performs better on the top-level classes. It achieves 87% when it makes prediction on all questions. Due to the inherent ambiguity of this problem, about 5%-10% questions are difficult to classify in one single class. Therefore, to distinguish hard questions and easy questions, the classifier is restricted to make a prediction only when it has high enough confidence. In this setting, it doesn't make a prediction on 15% to 20% of the questions, but the accuracy of the prediction is enhanced to 93%.

4.2 System Evaluation

Unfortunately, we didn't manage to process all the questions in TREC-10. The main reason is that we were too optimistic about the processing time given a huge set of data. Although the learning components we used, such as the POS tagger and shallow parser, are quite efficient, it still took a lot of time to finish all processing work. In addition, when there were many passages that were considered to be relevant, both Passage Retriever and Answer Selector took a long time to process.

From the 324 processed questions, we answered 108 correctly. In particular, 54 are in rank 1; 23 are in rank 2; 17 are in rank 3; 8 are in rank 4; 8 are in rank 5.

5 Conclusion

TREC-like question answering requires generating some abstract representation of the question, extracting (for efficiency reasons) a small portion of relevant text and analyzing it to a level that allows matching it with the constraint imposed by the question. This process necessitates, we believe, learning a large number of classifiers that need to interact in various ways and be used as part of a reasoning process to yield the desired answer.

This report summarizes some preliminary steps we took in this direction. We built several learning components to facilitate question answering. These components work pretty well independently but still fail short of the supporting a robust overall approach. Some of our future research directions include developing our unified approach further in several directions. These include using it to learn better representations for questions, more efficient syntactic and semantic building blocks and developing robust approaches for unification or reasoning when selecting answers to questions.

References

[1] A. Carlson, C. Cumby, J. Rosen, and D. Roth. The SNoW learning architecture. Technical Report UIUCDCS-R-99-2101, UIUC Computer Science Department, May 1999.

[2] C. Cumby and D. Roth. Relational representations that facilitate learning. In *Proc. of the International Conference on the Principles of Knowledge Representation and Reasoning*, pages 425–434, 2000.

[3] Y. Even-Zohar and D. Roth. A sequential model for multi class classification. In *EMNLP-2001, the SIGDAT Conference on Empirical Methods in Natural Language Processing*, pages 10–19, 2001.

[4] S. Harabagiu, D. Moldovan, M. Pasca, R. Mihalcea, M. Surdeanu, R. Bunescu, R. Girju, V. Rus, and P. Moreresc u. Falcon - boosting knowledge for answer engines. In *Proceedings of Text REtrieval Conference (TREC-9)*, 2000.

[5] L. Hirschman, M. Light, E. Breck, and J. Burger. Deep read: A reading comprehension system. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, 1999.

[6] A. Ittycheriah, M. Franz, W. Zhu, and A. Ratnaparkhi. Ibm's statistical question answering system. In *Proceedings of Text REtrieval Conference (TREC-9)*, 2000.

[7] E. F. T. Kim-Sang and S. Buchholz. Introduction to the CoNLL-2000 shared task: Chunking. In *Proceedings of CoNLL-2000 and LLL-2000*, pages 127–132, 2000.

[8] X. Li and D. Roth. Exploring evidence for shallow parsing. In *Proc. of the Annual Conference on Computational Natural Language Learning*, 2001.

[9] M. Munoz, V. Punyakanok, D. Roth, and D. Zimak. A learning approach to shallow parsing. In *EMNLP-VLC'99, the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 168–178, June 1999.

[10] V. Punyakanok and D. Roth. The use of classifiers in sequential inference. In *NIPS-13; The 2000 Conference on Advances in Neural Information Processing Systems*, pages 995–1001. MIT Press, 2001. Acceptance Rate: 25/514 (4.8%) Oral Presentations; 152/514 (29%) overall.

[11] D. Roth. Learning to resolve natural language ambiguities: A unified approach. In *Proc. of the American Association of Artificial Intelligence*, pages 806–813, 1998. Acceptance Rate: 143/475 (30%).

[12] D. Roth and W. Yih. Relational learning via propositional algorithms: An information extraction case study. In *Proc. of the International Joint Conference on Artificial Intelligence*, pages 1257–1263, 2001. Acceptance Rate: 197/796 (25%).

[13] R. Srihari and W. Li. Information extraction supported question answering. In *Proceedings of Text REtrieval Conference (TREC-8)*, 1999.